FIT2099 - Assignment 3
Design Rationale

Members:

Abhishek Shrestha
Amindu Kaushal

Objective: Explain how the system will work. Describe reasoning behind design decisions

Design Principles Used:

1. Avoid Excessive use of literals
2. Don't repeat yourself (DRY)
3. Classes should be responsible for their own properties
4. Reduce dependencies as much as possible
5. Declare things in the tightest possible scope
6. Delegation over inheritance
7. Command query principle
8. Liskov design principle

Thirsty Dinosaurs

To implement thirst, we have added an instance variable to store the current water level of the dinosaur which reduces by one each turn. And to keep track of unconsciousness we have added a separate counter for unconsciousness due to lack of water. We have **reused** the isConscious() method from Actor class by **overriding** it and updating it to return true if only both food and water levels are above 0. Regarding dinosaurs searching for water when thirsty, we have added a SeekWaterBehaviour class that implements the **Behaviour** interface and utilizes the SearchMap class(DrY) to look for a lake and returns a MoveActorToConsumeAction which moves the dinosaur towards the closest lake.

Pterodactyls

Since Pterodactyls should be able to fly, we have added Flight capability to easily distinguish between flying and walking. There are certain behaviours that are exclusive to Pterodactyls in comparison with other Dinosaurs, such as looking for a tree for resting when out of fuel, looking for a tree to lay an egg etc. So we have added PterodactylSeekTree behaviour which will be used for both resting and laying an egg, following the DrY principle. This behaviour is used in MateBehaviour if ready to lay an egg, and used in the Pterodactyl's playturn method if fuel is over. Since Pterodactyls can be eaten as a whole by Allosaurs, we have reused the AllosaurAttackBehaviour from assignment 2 to avoid repeating code, again following the DrY

principle. Additionally, we have made it possible for Pterodactyls to fly over vending machines, and prevented them from flying over walls.

Documentation of changes made in assignment 3

Dinosaur Classes

In addition to Stegosaurs, Brachiosaurs and Allosaurs, a new Pterodactyl class was added. In assignment 2, we added a VegetarianDinosaur and CarnivorousDinosaur class as subclasses from the abstract base Dinosaur class, so that we could add common actions such as FeedVegetarianAction to VegetarianDinosaur, and FeedCarnivoreAction to CarnivoreDinosaur following the Classes should be responsible for their own properties.
The new Pterodactyl was added as a child class of CarnivorousDinosaur and therefore it will be able to perform all the actions/behaviours that carnivorous dinosaurs can perform. This follows the DrY principle , since we only needed to add the actions/behaviours/attributes that were exclusive to Pterodactyls, and didn't need to repeat code for shared properties.

Dead Dinosaurs and Corpse

We have updated the Corpse class and added a corspePoints attribute to the class instead of creating four new classes for StegosaurCorpse, AllosaurCorpse, PterodactylCorpse and BrachiosuarCorpse because these classes were not necessary since the only common attribute was the amount of points that a carnivore will get by eating this corpse. This is following the ReD principle.
Doing this allowed us to ensure that Pterodactyls will only eat a portion of the corpse, so we were able to reduce the corpsePoints by a value 10, each time a Pterodactyl fed from it.

Vending Machine and functionality

We updated the vending machine class such that it will only store the prices of the items that are available for purchase instead of storing an instance of each item, therefore multiple unnecessary dependencies were eliminated, following the ReD principle. All purchases can be done through the BuyItemAction which allows the player to purchase an item (possibly shown by a menu) given there are enough ecoPoints. If so, that item will be instantiated and added to the player’s inventory. Additionally we have added a brand new PterodactylEgg item for purchase in the VendingMachine.

Players interaction with Dinosaurs

For players to be able to feed dinosaurs, we have created two classes, FeedVegetarianAction and FeedCarnivoreAction since there are two types of dinosaurs in the game. Since one type of dinosaur eats meat,eggs and carnivore meal kits while the other type eats fruits and vegetarian meal kits, having two separate classes would make the code much neater, more structured and allows us to avoid excessive use of literals. Using two classes for feeding instead of the initial plan of having a separate feed action class for each dinosaur allowed us to reduce dependencies (ReD), and the previous design would not be efficient in design terms, since each time a new dinosaur is added to the game, a new feed action class will have to be created, whereas as having two flexible classes is better for future updates to the game.
Regarding controlling the population of Stegosaurs, we have re-used the existing AttackAction for attacking Stegosaurs, to avoid repeating code (DrY) and to minimize dependencies (ReD)

Searching the map

Since we often found ourselves searching for either an item(fruits, corpses), ground(lake, bushes, trees) or an actor(pterodactyl, stegosaur) we have added a seperate class called SearchMap that iterates through the entire map and looks for the target object. This process of looking through the map has been used in the implementation of multiple design features such as

- SeekFoodBehaviour : Different types of dino's looking for food they eat
- SeekWaterBehaviour : Dinos searching for water when thirsty
- AllosaurAttackBehaviour : Allosaurs seeking for nearby Pterodactyls/Stegosaurs
- PterodactylSeekTreeBehaviour : Pterodactyls looking for trees for resting/laying eggs

Therefore using this class allowed us to remove an extensive amount of repeating code(DrY)

Dinosaur seeking food

Previously we had two separate classes for seeking food, one was SeekMeat and the other was SeekFruit, to reduce dependencies(ReD) we decided to combine these two classes and create a SeekFoodBehaviour class which utilizes the SearchMap described above to search the map for the right type of food for each dinosaur. This combination also allowed us to reduce some repeating code that was used in both classes (DrY). Furthermore, this class will be more flexible for more additions in edible items to the game, aside from just fruits and meat. The SeekFoodBehaviour implements the **Behaviour** interface.

Priorities

When a dinosaur has multiple needs in the same turn, we have set a priority system as explained below:
For Stegosaurs/Brachiosaurs

1. Mating
2. Laying egg
3. Thirst
4. Hunger

For Allosaurs

1. Mating
2. Laying Egg
3. Attacking (regenerates health points)
4. Thirst
5. Hunger

For Pterodactyls

1. Mating
2. Laying egg
3. Moving towards tree for resting (regaining ability to fly)
4. Moving towards tree for laying egg
5. Thirst
6. Hunger

We have decided that this is the most logical approach for priorities, and also the best method for their survival. For Allosaurs, after mating/laying egg the next best priority is attack(Stegosaur/Pterodactyl), we have done that to make the game more interesting and to increase the importance of the danger of Pterodactyls walking instead of flying, since they would be eaten completely by Allosaurs. For Pterodactyls, due to the previously mentioned reason, we have made resting the next priority after mating/laying eggs.

Table mapping display characters to important actors/items:

Display char	Object
a	Allosaur (Baby)
A	Allosaur (Adult)
b	Brachiosaur (Baby)
B	Brachiosaur (Adult)
s	Stegosaur (Baby)
S	Stegosaur (Adult)

p	Pterodactyl (Baby)
P	Pterodactyl (Adult)
q	StegosaurEgg
w	BrachiosaurEgg
e	AllosaurEgg
y	PterodactylEgg
v	VegetarianMealKit
c	CarnivoreMealKit
C	Corpse
Z	LaserGun